|  |  |
| --- | --- |
| **LCA in C++** | |
| #include <iostream>  using namespace std;  // Define Node structure for BST  struct Node {  int key;  Node \*left, \*right;    Node(int item) {  key = item;  left = nullptr;  right = nullptr;  }  };  // Function to find LCA of two nodes in BST  Node\* getLCA(Node\* node, int n1, int n2) {  if (node == nullptr) {  return nullptr;  }    // If both n1 and n2 are smaller than root, then LCA lies in left subtree  if (node->key > n1 && node->key > n2) {  return getLCA(node->left, n1, n2);  }    // If both n1 and n2 are greater than root, then LCA lies in right subtree  if (node->key < n1 && node->key < n2) {  return getLCA(node->right, n1, n2);  }    // Otherwise, root is LCA  return node;  }  int main() {  // Create the BST  Node\* root = new Node(6);  root->left = new Node(3);  root->right = new Node(8);  root->right->left = new Node(7);  root->right->right = new Node(9);    // Find LCA of nodes 3 and 7  Node\* lca = getLCA(root, 3, 7);  cout << "LCA of 3 and 7 is: " << lca->key << endl;    return 0;  } | BST Structure: 6  / \  3 8  / \  7 9 🔍 Goal: Find LCA of ****3**** and ****7****🧾 Dry Run Table:  | **Function Call** | **Node Key** | **Comparison (n1=3, n2=7)** | **Decision** | **Return Value** | | --- | --- | --- | --- | --- | | getLCA(root, 3, 7) | 6 | 3 < 6 AND 7 > 6 | Split → current node is the LCA | 6 |  🖨 Final Output: LCA of 3 and 7 is: 6 |
| LCA of 3 and 7 is: 6 | |